



POT/GB2004/001583



INVESTOR IN PEOPLE

GB04/1583

**PRIORITY  
DOCUMENT**  
SUBMITTED OR TRANSMITTED IN  
COMPLIANCE WITH RULE 17.1(a) OR (b)

The Patent Office  
Concept House  
Cardiff Road  
Newport

South Wales

NP 2004 001583 MAY 2004

WIPO

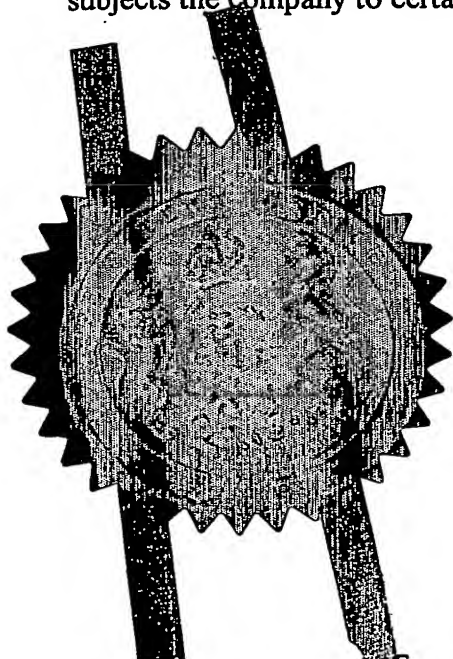
PCT

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.



Signed

*W. Evans*

Dated 30 April 2004

Patent Act 1977

(6)

15 APR 2003

LONDON

15 APR 2003

The Patent Office

Cardiff Road

Newport

Gwent NP10 8QQ

**Request for grant of a patent**

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

1. Your reference

A30304

2. Patent application number

*(The Patent Office will fill in this part)*

0308708.7

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

BRITISH TELECOMMUNICATIONS public limited company  
81 NEWGATE STREET  
LONDON, EC1A 7AJ, England  
Registered in England: 1800000

Patents ADP number *(if you know it)*

1867002 ✓

If the applicant is a corporate body, give the country/state of its incorporation

UNITED KINGDOM

4. Title of the invention

A COMPUTER SYSTEM

5. Name of your agent *(if you have one)*

"Address for Service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

BT GROUP LEGAL  
INTELLECTUAL PROPERTY DEPARTMENT  
HOLBORN CENTRE  
120 HOLBORN  
LONDON, EC1N 2TE

Patents ADP number *(if you know it)*~~1867001~~

8591919001

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country

Priority application number  
*(if you know it)*Date of filing  
*(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application

Date of filing  
*(day/month/year)*8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

- a) any applicant named in part 3 is not an inventor, or
- b) there is an inventor who is not named as an applicant, or
- c) any named applicant is a corporate body.

*(See note (d))*

YES

**Patents Form 1/77**

9. Enter the number of sheets for any of the following items you are filing with this form.  
Do not count copies of the same document

Continuation sheets of this form

Description 11

Claim(s) 4

Abstract 1

Drawing(s) 6

10. If you are also filing any of the following, state how many against each item

**Priority Documents**

Translations of priority documents

Statement of inventorship and right to grant of a patent (Patents Form 7/77)

Request for preliminary examination and search (Patents Form 9/77) YES

Request for substantive examination (Patents Form 10/77)

Any other documents (please specify)

11. I/We request the grant of a patent on the basis of this application.

Signature(s)

Date:

15 April 2003

LLOYD, Barry George William, Authorised Signatory

12. Name and daytime telephone number of person to contact in the United Kingdom

Rod Hillen

020 7492 8140

**Warning**

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

**Notes**

- If you need help to fill in this form or you have any questions, please contact the Patent Office on 0645 500505.
- Write your answers in capital letters using black ink or you may type them.
- If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.
- If you have answered 'Yes' Patents Form 7/77 will need to be filed.
- Once you have filled in the form you must remember to sign and date it.
- For details of the fee and ways to pay please contact the Patent Office.

## A Computer System

This invention relates to a computer system, in particular to a computer system having a plurality of components which can be initialised.

5

Despite advances in computer technology, the components of a computer system can have deficiencies which cause faults to develop over time. For example, many applications, such as those written in the language "C" contain memory leaks, wherein the application does not release memory after it has finished using it. Such faults can cause a component to hang or crash, thereby hindering the efficient operation of the system. It is known to cure or prevent a fault by initialising the faulty or potentially faulty component, for example by replacing at least some of the software from the component, by re-starting or re-booting the component, or by re-setting one or more control parameters of that component. However, the initialisation process will often disrupt the normal operation of the component, at least during the period in which initialisation is taking place. During this period, other components may be required to perform an additional number of tasks to compensate for the tasks that are not being performed by the disrupted component. Such a re-distribution of tasks may reduce the efficiency of the system as a whole.

20

According to one aspect of the present invention, there is provided a computer system having a plurality of components that can be initialised, wherein each component is configured to produce status data from which the level of need for that component to be initialised can be inferred, and wherein at least one component is configured to: receive status data from other components; make a comparison using the status data received from respective components; in dependence on the comparison, select one or more components for initialisation; and, issue initialisation instructions to the selected component(s).

30

By making a comparison between the status data from different components, the relative need for different components to be initialised can be evaluated, allowing a component with a high need for initialisation to be selected over a component with a lower need. Thus, rather than considering in isolation whether a component needs to

~~be initialised, the present invention allows the initialisation of components to be~~  
prioritised, thereby reducing the amount of disruption to the system as a whole.

The components may be hardware components, or alternatively, the components  
5 ~~may be software components, running on at least one (hardware) computer device.~~

In one embodiment, the components are software components each running on a  
respective computer device, the initialisation of a component causing the device on  
which it runs to be re-booted or otherwise initialised.

- 10 The status data received from a component may include a plurality of status values, such as the amount of free memory in a memory location, the frequency with which a processor is accessing a memory location, or other historical or current data which can be used to determine the need for initialisation of the component.
- 15 However, to simplify the processing to be carried out by a component receiving the status data, the status data will preferably be in the form of an initialisation parameter. This will reduce the need for a component to evaluate and/or cross reference different data originating from the same component in order to infer the need for that component to be initialised. Furthermore, if the need for a component
- 20 to be initialised can be expressed simply in terms of a variable, such as the elapsed time since the last initialisation, the respective values of that variable for different components can be directly compared against one another.

The level of need for a component to be initialised can be positive or negative. Thus  
25 an initialisation parameter may indicate the importance or urgency for a component to be initialised, but an initialisation parameter may alternatively indicate the importance for a component not to be initialised, for example if that component is carrying out an essential task.

- 30 Preferably, each component will be configured to execute an initialisation routine when the initialisation parameter for that component reaches a threshold value, in which case that component will behave as an initialising component, sending a request message to other (recipient) components, the request message requesting

respective initialisation parameters from the recipient components. Hence, when the need for a component to be initialised becomes sufficiently high, that component can at least attempt to find out the need for other components to be initialised. This allows the amount of information which a component needs to retain about the state of other components to be reduced, since a component can when required obtain such information by transmitting a request message. Furthermore, if each component can request initialisation parameters from other components, the need for the initialisation requirements of different components to be stored at a central location is reduced.

10

The threshold value for each component may be the same or different, and may be set by a system administrator in dependence on the importance or the number of tasks performed by a component.

- 15 The initialisation routine carried out by an initialising component preferably includes the further steps of: comparing the initialisation parameters received from other components with the initialisation parameter for the initialising component; and, in dependence on the comparison, making a self-initialisation decision. In this way, each component can take into account the need of other components to initialise
- 20 before taking a self-initialisation decision.

In a preferred embodiment, each component is configured to select components for initialisation, and to issue initialisation instructions to the selected components. Thus, in this embodiment, even if one component is unable, due to a fault or

25 otherwise, to act as an initiating component and select components for initialisation, another component of the system will eventually be triggered to act as an initiating device (for example because the time since the last initialisation of that component has exceeded a threshold value). This fault tolerance will allow the system to maintain itself efficiently, since the task of selecting which components to initialise

30 can in effect be distributed across the system.

The invention will now be further described, by way of example, with reference to the following drawings in which:

Figure 1 shows schematically a computer system according to the invention;

Figure 2 shows a more detailed view of a computer device of Figure 1;

Figure 3 is a flow chart showing steps carried out at an initiating device;

5 Figure 4 is a flow chart showing steps carried out at a recipient device; and,

Figure 5 shows schematically three stages involved in the selection of devices for initialisation; and,

Figure 6 is a UML sequence diagram illustrating the communication and relationship between objects in the computer system.

10

In Figure 1, there is shown a computer system 10 which includes a plurality of computer devices 22, each with a respective software component 18 running thereon, the computer devices being interconnected by a plurality of data links 14. Tasks or jobs are dynamically allocated to the respective computer devices 22 by a  
15 server 16, such that the tasks or jobs allocated to one computer device depend on the tasks or jobs being performed by at least some of the other computer devices. In this way, the computer system 10 as a whole can operate more efficiently than it would do in the absence of such coordination between the computer devices 22.

20 Each computer device is able to generate an initialisation parameter that is indicative of the need (or equivalently the urgency or desirability) for the computer device 22 to be initialised. When the initialisation parameter of a computer device 22 reaches a threshold value, that computer device acts as an initiating device 22a, and transmits a request message 20 over the data links 14 to the other computer devices 22,  
25 which devices then act as recipient devices 22b. The request message requests from the recipient devices 22b the current values of their respective initialisation parameters.

The initiating device 22a is configured to compare the received initialisation  
30 parameters from the recipient devices 22b together with its own initialisation parameter and to issue initialisation instructions to itself and/or one or more recipient device(s) 22b in dependence on the relative values of the initialisation parameters. Because the initialisation parameter of a computer device 22 is a measure of the need

for that device to be initialised, the computer devices 22 most needing initialisation can be selected in favour of computer devices least requiring initialisation. Thus, those computer devices 22 which suffer disruption brought about by their initialisation can be chosen so as to reduce the overall disruption to operation of the system 10.

In effect, the recipient devices 22b participate in an "auction", the initialisation parameters representing "bids" for one or more available initialisation instructions. The initiating device 22a can be viewed as an "auctioneer" device, allocating initialisation instructions to the devices which place the highest bids: that is, those devices (including the initiating device) which are most in need of initialisation. Although in this example the bids are "sealed", in that the bid placed by one device is independent of bids placed by other devices (and different devices can "bid" at the same time), alternative embodiments are possible where a feedback routine is executed by the devices participating in the auction, such that the bid placed by one device is dependent on bids placed by another device.

The "auction" procedure or protocol can be summarised with reference to Figure 5 in terms of three stages labelled I to III. In stage I the initiating (auctioneer) device 22a announces its intention to hold an auction, by broadcasting a request message 11 to other recipient devices 22b. In stage II, devices not participating in another auction return bids 13 to the auctioneer device, whereas devices (indicated by shading) already participating in another auction with another auctioneer device (not shown) ignore the request message 11. The self-elected auctioneer device 22a also places a bid in its own auction. In stage III, after the auctioneer 22a has received and processed the bids, it informs the participants of the auction (non-shaded recipient devices 22b) of the "winning" devices which may be initialised (or equivalently another maintenance task can be initiated). To inform the participants, the auctioneer sends a message 15 to the participants, which message includes: the number of devices participating in the auction; the identity of the winning devices; and, the total number of devices participating in the auction.

Each computer device 22 may be housed in a respective housing or casing, the data links 14 being formed by cables which extend between the housings of the respective computer devices. In such a situation, the computer devices 22 may be situated in different geographical locations. Alternatively, the computer devices 22 may be located in a common housing, and even for example on the same circuit board.

A more detailed view of one of the computer devices 22 of Figure 1 is shown in Figure 2. The computer device 22 has a processor circuit 26 which includes a memory 31 and a processor 27 for respectively storing and executing the software component 28, as well as other software which may be running on the computer 22.

The software component 28 includes an operating system module 40 for controlling the operation of the computer device 22, and an application programme 42 for performing tasks. A monitoring stage 34 is provided for monitoring one or more status indicators indicative of the likelihood of a fault or potential fault with the computer device 22, and for producing an initialisation parameter in dependence on the monitored status data. The monitoring stage 34 may monitor the amount of memory space in the memory 31 that is deemed available by the operating system module 40. Thus, if the application programme 42 fails to release all the memory it has finished using, this "leaky programme" malfunction (or "bug") which could possibly lead to a fault can be monitored. When the amount of available memory drops below a threshold value, this can be classified by the monitoring stage as a fault or potential fault, and the monitoring stage can, in response to this fault or potential fault, issue an initialisation parameter indicating a high need for the operating system module 40 and/or the application 42 to be initialised.

However, in a preferred embodiment, the monitoring stage includes a timer 35 for measuring the elapsed time since the last initialisation (preferably the time since the last initialisation of the operating system module 40), the initialisation parameter being representative of this elapsed time. It is appreciated that because computer devices are likely to behave in a non-ideal way due to malfunctions such as the "leaky programme" malfunction, and because such malfunctions are likely to cause a

deterioration in the operational capacity of a computer device over time, the elapsed time since the last initialisation can provide a useful measure of the need for a computer device to be initialised.

- 5 The monitoring stage may take into account the tasks being performed by the device on which it is running, more important tasks having a greater weighting on the initialisation parameter than tasks of a lesser importance. In one embodiment, the monitoring stage monitors whether the device on which it runs is performing one or more of a predetermined number of task, and will only issue a non-zero initialisation  
10 parameter to the if such a task is not being performed.

The initialisation may involve a re-boot (or equivalently a re-start) of the computer device 22, or alternatively the initialisation may simply require the application 42 to be killed and re-started, without re-booting the entire computer device 22. As a  
15 further alternative, the initialisation may require the amount of memory deemed in use by the application 42 to be re-set, for example to zero.

The initialisation parameter generated by the monitoring stage 34 is passed to an initialisation controller 32. The initialisation controller 32 is configured to determine  
20 whether the initialisation parameter (in this example, the elapsed time since the last initialisation) from the monitoring stage 34 is above a threshold value. If the initialisation parameter is above a threshold value, the computer device 22 then acts as an initiating device 22a as indicated in Figure 3, and executes an initialisation routine. In the initialisation routine, the initialisation controller 32 of the initiating  
25 device 22a transmits at an output stage 36 one or more requests for an initialisation parameter to the other computer devices 22 (acting as recipient devices 22b).

The initiating device 22a need not know the identity of the recipient devices, nor need the initiating device know the number of existing recipient devices. Therefore,  
30 the request for initialisation parameters from the initiating device 22a will preferably be sent in the form of a broadcast message, normally on a well-known port, and normally without being addressed to any specified recipient devices 22b. The request message will preferably contain: a time stamp indicating the time at which

the message was sent; the name, address, and/or other label which identifies the initiating device 22a from which the request message originates; information indicative of a port number on which a reply with an initialisation parameter should be sent; and a unique identifier for the request message.

5

For each computer device 22, there is provided a listening stage 36 configured to monitor data arriving at that device 22, so as to detect if a request for an initialisation parameter has been transmitted by another device 22 (the initiating device 22b), preferably on the chosen well-known port. When a request from an  
10 initiating device 22a is detected by the listening stage 36 of a recipient device 22b, the request is passed to the initialisation controller 32 of that device, in response to which the initialisation controller 32 executes a response routine (steps carried out by the recipient device 22b are shown in Figure 4).

15 As part of the response routine, the initialisation controller 32 of the recipient device 22b determines whether the recipient device 22b is involved in an initialisation procedure with another initiating device. If so, the initialisation controller 32 does not respond to the most recent request for an initialisation parameter. Otherwise, the initialisation controller 32 of the recipient device 22b will access status data or an  
20 initialisation parameter from the monitoring stage 34 of that device. The monitoring stage 34 determines if the recipient device 22b is busy performing one of a number of tasks, and if so, passes a busy signal to the initialisation controller 32. If the recipient device 22b is not busy, the monitoring stage 34 passes to the initialisation controller 32 an initialisation parameter indicative of the elapsed time since the last  
25 initialisation at the recipient device 22b. The initialisation controller 32 of the recipient device 22b then transmits the initialisation parameter to the initiating device 22a, unless a busy signal is received from the monitoring stage 34, in which case the initialisation controller 32 transmits the busy signal to the initiating device 22a in the form of an initialisation parameter which indicates the elapsed time since the last  
30 initialisation of the recipient device 22b is zero seconds. This busy signal will allow the initiating device 22a to treat the recipient device 22b as having the lowest possible priority for initialisation.

The listening stage 36 of the initiating device 22a captures any initialisation parameters received from recipient devices 22b in response to the request(s), and forwards these received initialisation parameters to the initialisation controller 32 of the initiating device 22a. The initialisation controller 32 then performs a comparison step using the initialisation parameters received from recipient devices 22b and the self-generated initialisation parameter from the monitoring stage 34 of the initiating device 22a. The comparison step involves ranking at least some of the initialisation parameters from different devices 22 in dependence on their respective magnitudes, and selecting a proportion (such as 10%) of the initialisation parameters having the highest values. An initialisation instruction is then sent to the respective computer devices 22 whose initialisation parameter falls within that proportion. (If one of those initialisation parameters used in the comparison steps originates from the initiating device 22b, the initialisation controller 32 of the initiating device 22b will issue a self-initialisation instruction to the operating system module 40 and/or the application 42 running on the initiating device). In this way, the devices 22 of the computer system 10 which most urgently require initialisation can be selected.

The comparison step may also involve taking into account the total number of responses to the request message, as well as the number of idle devices (i.e. devices which return a non-zero value initialisation parameter). Thus, 10% of idle devices may be selected for initialisation.

The number or percentage of devices chosen for initialisation may be set by an administrator or the server 16, and may depend on the predicted workload of the system (the workload may be predicted by analysing historical patterns in the way in which the server 16 allocates tasks to the respective device, or by inspecting a log of tasks awaiting allocation by the server 16). By predicting the likely workload on the system, it is possible to estimate how many devices or what percentage of devices can be re-booted without endangering the operation of the system as a whole. In this way, the re-boot times of the devices 22 are co-ordinated so as to make it likely that at any one time, there will be enough working devices to handle job requests arriving at the system 10.

The initialisation instructions from the initiating device 22a will preferably be sent as a common initialisation message to the recipient devices 22b, such that the recipient devices 22b receive the same message. The initialisation message will then contain information identifying which recipient devices 22b should be initialised, so that the  
5 initialisation controller of an identified recipient devices 22b can act on the initialisation message. The initialisation message will preferably also contain information indicating the number of recipient devices 22b from which an initialisation parameter was received and the number of recipient devices 22b identified for initialisation.

10

The initialisation instruction for a recipient device 22b will preferably instruct the recipient device 22b to initialise within a time period. However, before instigating an initialisation, the initialisation controller 32 of the recipient device 22b will interrogate the monitoring stage 33 to determine if the recipient device 22b is performing any of  
15 a number of tasks (and optionally whether the tasks(s) will be continued beyond the time period allowed for in the initialisation request). If so, the recipient device 22b is deemed to be busy, and the initialisation controller of the recipient device 22b ignores the initialisation instruction. Optionally, the initialisation controller 32 of the recipient device 22b can be configured to carry out an initialisation in response to the  
20 initialisation instruction if the elapsed time since the last initialisation exceeds a threshold value, regardless of whether the recipient device 22b is busy or idle.

In Figure 6 there is shown a UML sequence diagram illustrating the communication and relationship between software objects in the computer system (each object or  
25 process is represented as a box in Figure 6). Each device 22 that can potentially take part in the auction runs a listening process 61 indicated by the RunListener time line. This process waits for communication from other devices. When a device has reached the threshold necessary to hold an auction, a process 63 indicated by the RunAuction time line starts. This RunAuction process 63 creates a BidManager  
30 process 65 and informs it which communication port to use. The BidManager process 65 creates a BidCollector process 67 and passes it details of the communications port on which to receive bids. The BidManager process 65 contacts the RunListener running in each devices by sending a message 69. When the RunListener process 61

of a device 22 receives such a message 69 it creates a BidMaker process 71 for that device supplying device name and communications port number. The BidMaker process 71 sends a placeBid message 73 to the BidCollector 67. The BidManager also creates a BidEvaluator process 75 which calculates the winning bids i.e., which  
 5 selects devices for initialisation. Winners (i.e., devices selected for initialisation) are informed via the BidCollector process and the BidMaker process. The sequence shown in Figure 6 is preferably implemented using (or written in) the "Java" language.

- 10 The embodiment described above is suitable for a collection of computers. Because the protocol in this embodiment does not need to know the membership details of the collection it is able to continue to work in the face of partition of the collection. As an example of this, if we consider 2 islands of computers, for example 50 in each, linked by a network connection into a single pool where the "auction protocol" of the  
 15 invention is active, scheduling of maintenance or initialisation will take place as and when required. Should the network connection fail, each island can continue scheduling maintenance within itself, until the connection is restored when the whole pool can participate again.
- 20 In another embodiment (not shown), the software components 28 run on a common computer device under the control of a common operating system. An example of this is a server running different programs for different users. In such a situation, the software components 28 will not normally include the operating system, such that the operating system can continue to run even when one or more of the components  
 25 28 are initialised.

As can be seen from the above description, the present invention will be particularly useful in a system having components which work together as a group and which each require initialisation at time intervals, allowing components to initialise without  
 30 excessively disturbing the group dynamics of the system.

## CLAIMS

1. A computer system having a plurality of components that can be initialised,  
5 wherein each component is configured to produce status data from which the level of  
need for that component to be initialised can be inferred, and wherein at least one  
component is configured to: receive status data from other components; make a  
comparison using the status data received from respective components; in  
dependence on the comparison, select one or more components for initialisation; and,  
10 issue initialisation instructions to the selected component(s).

2. A computer system as claimed in claim 1, wherein the or each component  
configured to make a comparison using status data is configured to use its own  
status data in addition to the received status data when making the comparison.

15

3 A computer system as claimed in claim 1 or claim 2, wherein the  
components are software components, and wherein the system includes at least one  
computer device on which, in use, the software components are run.

20 4 A computer system as claimed in any of claims 1 to 3, wherein the status  
data is in the form of an initialisation parameter.

5 A computer system as claimed in claim 4, wherein each component is  
configured to execute an initialisation routine when the initialisation parameter for  
25 that component reaches a respective threshold value, the initialisation routine  
including the step of transmitting a request for an initialisation parameter to other  
components.

6 A computer system as claimed in claim 5, wherein the initialisation routine  
30 includes the further steps of: receiving initialisation parameters from at least some of  
those other components; comparing the received initialisation parameters with the  
initialisation parameter for that component; and, in dependence on the comparison,  
making a self-initialisation decision.

7        A computer system as claimed in any of claims 4 to 6, wherein each component includes a timer module for registering the elapsed time since the previous initialisation of that component, and wherein for each component, the  
5        initialisation parameter is determined at least in part in dependence on the elapsed time registered by the timer module.

8        A computer system as claimed in any of claims 4 to 7, wherein each component is configured to produce an initialisation parameter that is at least in part  
10       dependent on whether the component is performing one of a number of predetermined tasks.

9        A computer system as claimed in any of claims 3 to 8, wherein the computer system includes a plurality of interconnected computer devices, each of which is  
15       housed in a respective housing, and wherein each device has, in use, a respective software component running thereon.

10.      A computer system as claimed in claim 9, wherein the software components each include a respective operating system module for operating the computer device  
20       on which the respective software component is running.

11.      A computer system as claimed in claim 9 or claim 10, wherein each component is configured to initiate a re-boot routine upon receipt of an initialisation instruction, the re-boot routine being configured to re-boot the computer device on  
25       which the software component is running.

12.      A computer system as claimed in claim 11, wherein the re-boot routine includes the step of determining if the computer device is performing a predetermined task or one of a number of predetermined tasks, and only to permit the re-booting of  
30       the computer device if the computer device is not performing such a task.

13. A computer system as claimed in any of claims 3 to 8, wherein the components in use run on a common computer device, under the control of a common operating system.

5 14. A computer system as claimed in any of claims 3 to 13, wherein each component, upon receipt of an initialisation instruction, is configured such that the component is killed and subsequently restarted.

15. A computer system as claimed in any preceding claim, including a computer  
10 device configured to allocate tasks to the components, such that a task allocated to one component is dependent on the task or tasks being performed by at least some of the other components.

16. A method of initialising the components of a computer system, the method  
15 including the steps of: using status data from a plurality of components to determine for each component, the need for that component to be initialised relative to the need for at least one other component to be initialised; and, initialising at least some of the components in dependence on their so determined relative need.

20 17. A computer device having, in use, a software component running thereon, the software component being configured to: receive status data from a plurality of other components; make a comparison using the status data received from the respective components; in dependence on the comparison, select one or more components for initialisation; and, issue initialisation instructions to the selected  
25 component(s).

18. A computer device as claimed in claim 17, wherein the software component running thereon is configured to produce status data from which the need for that component to be initialised can be inferred.

30

19. A computer device as claimed in claim 18, wherein the software component is configured to compare its self-produced status data with status data received from

at least one other component when selecting one or more components for initialisation.

20. A computer programme, the computer programme being configured, when  
5 loaded on a computer device, to: receive status data from a plurality of software components; make a comparison using the received status data from the respective components; in dependence on the comparison, select one or more of the software components for initialisation; and, issue initialisation instructions to the selected software component(s).
- 10 21. A computer programme as claimed in claim 20, wherein the computer programme is configured to issue initialisation instructions for re-booting the computer device or another computer device.
- 15 22. A computer programme product stored on a computer-useable medium, the computer programme product having a program stored thereon as claimed in claim 20 or claim 21.

## ABSTRACT

## A computer system

- 5 This invention relates to a computer system, in particular to a computer system having a plurality of components which can be initialised. There is provided a computer system having a plurality of components that can be initialised, wherein each component is configured to produce status data from which the level of need for that component to be initialised can be inferred. At least one component is
- 10 configured to: receive status data from other components; make a comparison using the status data received from respective components; in dependence on the comparison, select one or more components for initialisation; and, issue initialisation instructions to the selected component(s). By making a comparison between the status data from different components, the relative need for different components to
- 15 be initialised can be evaluated, allowing a component with a high need for initialisation to be selected over a component with a lower need.

Figure 5

1/6

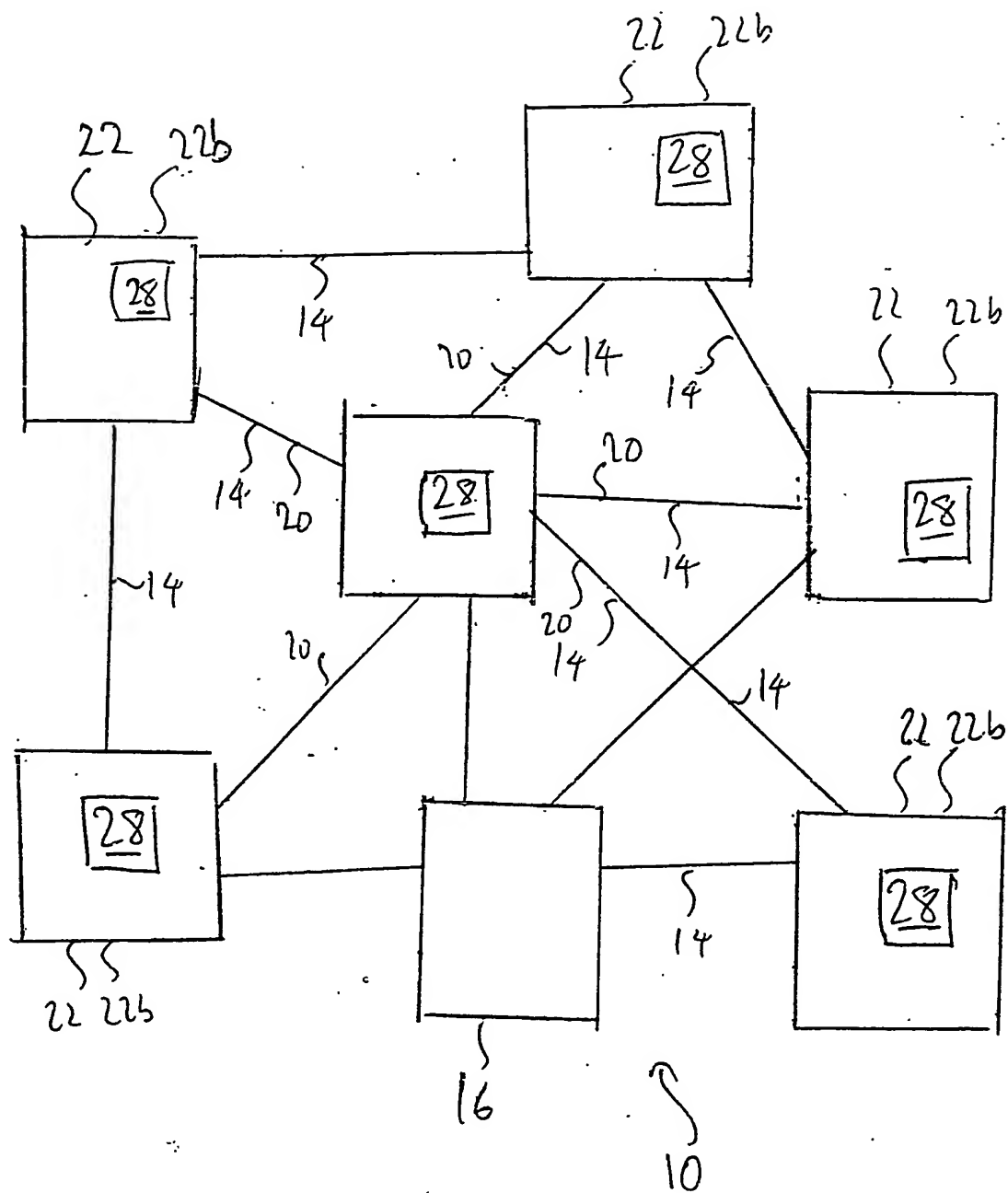


Fig. 1

2/6

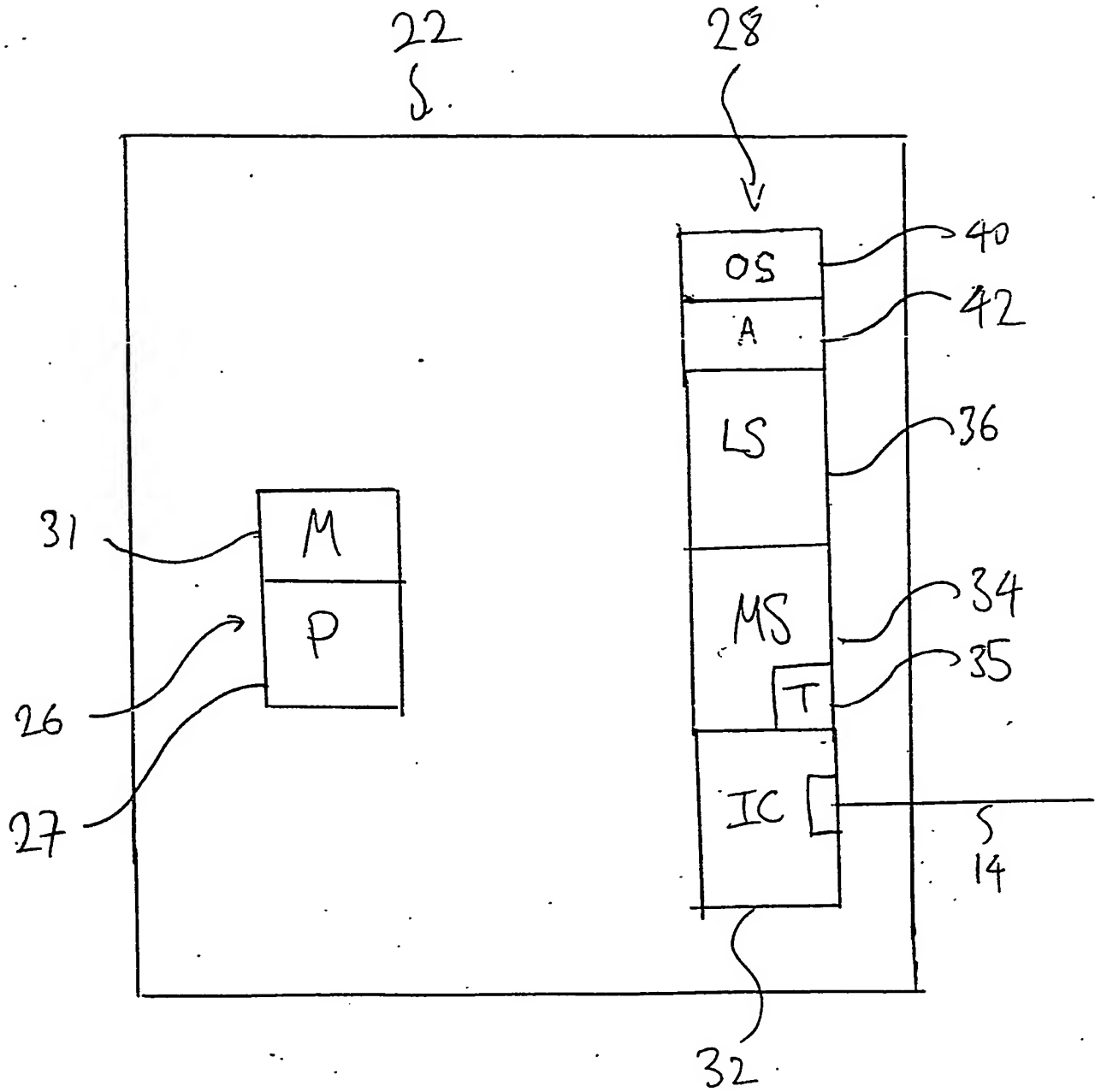


Fig. 2

3/6

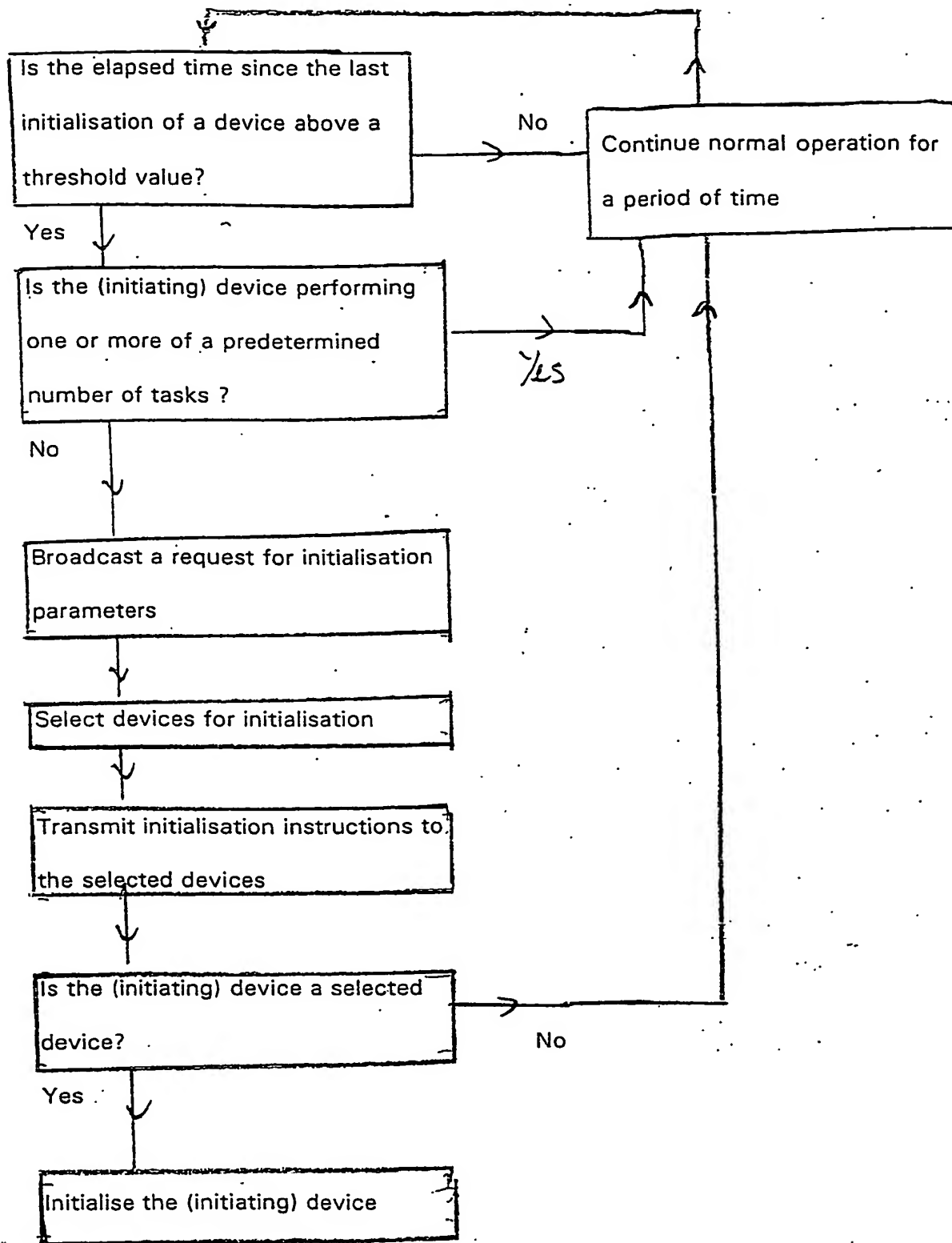


Figure 3

4/6

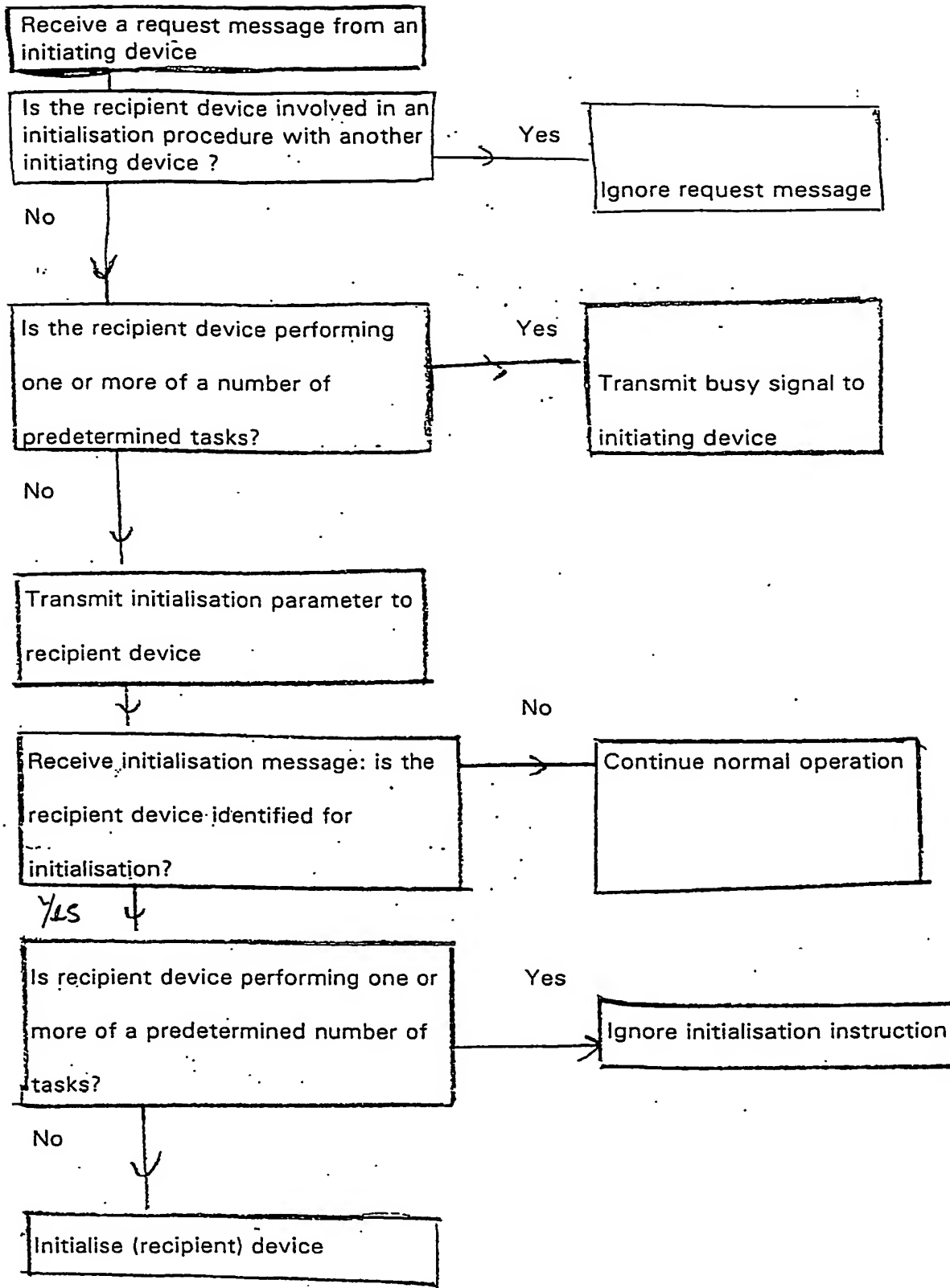


Figure 4

5/6

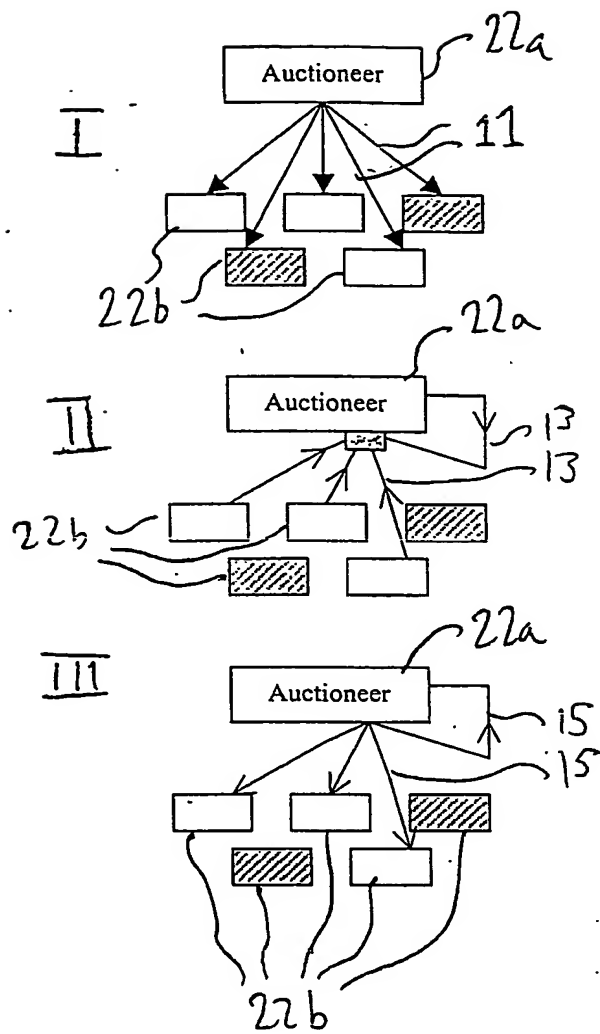


Fig. 5

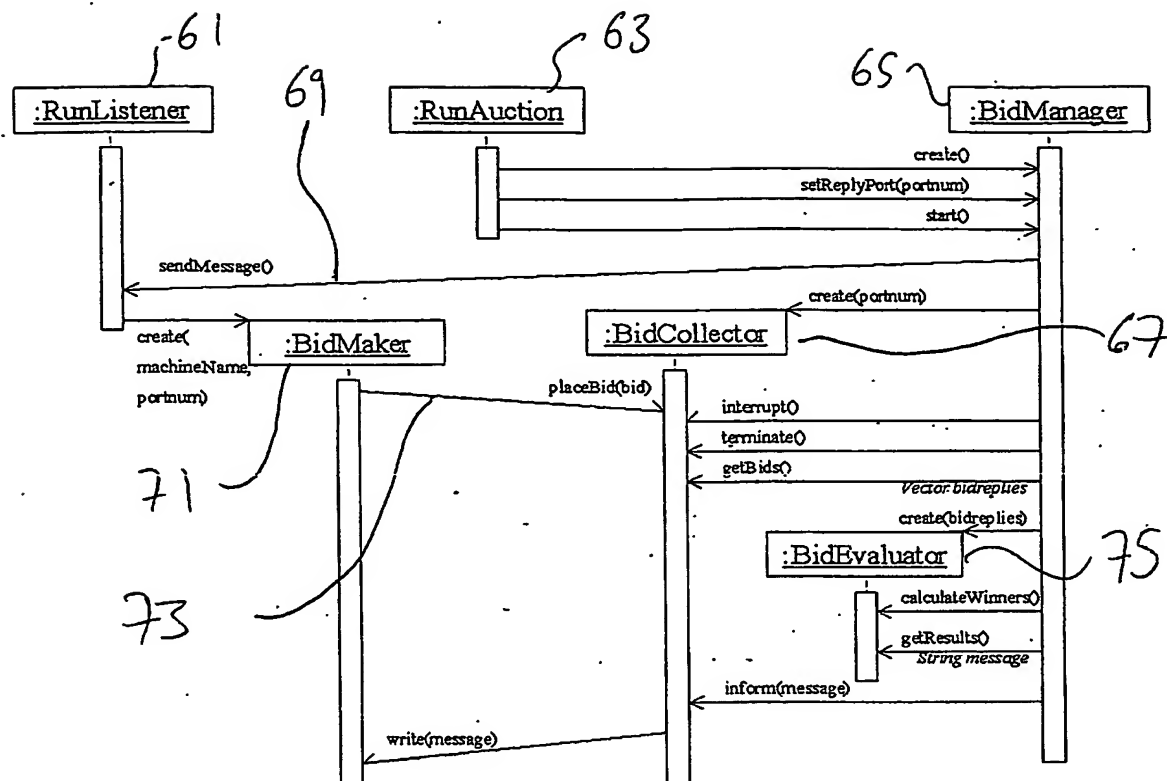


Figure 6



PCT/GB2004/001583



This Page is inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLORED OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☒ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☒ REPERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images  
problems checked, please do not report the  
problems to the IFW Image Problem Mailbox**